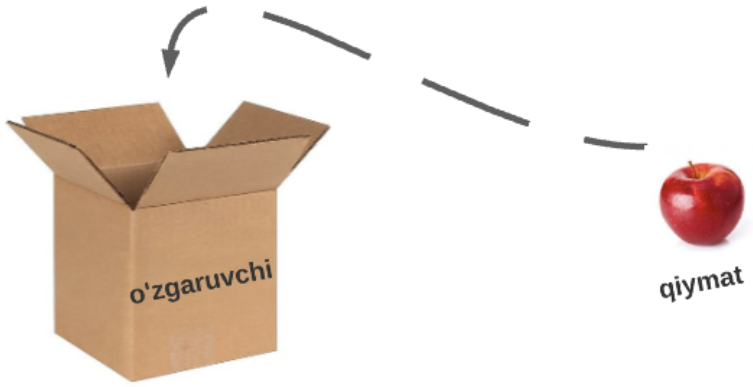
O'zgaruvchilar va ifodalar

Pythonda eng muhim tushuncha bu -- **o'zgaruvchi** (inglizchada **variable**).
O'zgaruvchilarni tushunish o'ta muhim. Tushunishga bir muncha yengillik bo'lishligi uchun, quyidagi
qiyoslashni olsak.

Olmani qutiga yuklang

Aytaylik, bizga olma va quti berilgan. Pastdagi grafga qarang. Muammo (problem solving) shundan iboratki,
olmani yashikka yuklash (solish) kerak. Xo'sh qanday hal qilamiz bu muammoni? Bu juda oson **Olmani qo'limiz bilan olib, uni yashikka yuklaymiz.**
Bo'ldi muammo hal.



Xo'sh bu yuklashni kompyuterda qanday hal qilamiz? To'g'rirog'i, kompyuterga qanday qilib `olmani qutiga yukla` deb ko'rsatma beramiz?

Kompyuterda quyidagicha ko'rsatma beramiz:

```
quti = 'olma'
```

Bu yerda '=' belgisi `yuklash` ni bildiradi. Ona tilimizga tarjima qilsak: 'olmani qutiga yukla' deb ko'rsatma
berayapmiz. E'tibor qilishimiz kerakki, biz chapdan o'ngga qarab o'qiyapmiz.

Bu yerda `olma` **qiymat** deb ataladi. `Quti` esa **o'zgaruvchi** deb ataladi.
O'zgaruvchi deyilishiga sabab shundaki, qutiga biz boshqa narsa ham solishimiz mumkin, masalan

- Tarvuzni qutiga yukla
- Qo'yni qutiga yukla

Bunda qutiga nima yuklanishiga qarab quti o'zgarayapti. Olma yuklansa olma bo'ladi, qo'y yuklansa qo'y bo'ladi va hokazo.

Avvalgi darsimizda biz `"Salom, Python!"` ni konsolga chiqargan edik. Unda `print` ko'rsatmasidan
foydalangan edik

```
print("Salom, Python!")
```

Bu safar ushbu misolni o'zgaruvchi yordamida qilamiz. Dehqonchasiga,

- (1) `Salom, Python` qiymatini `x` nomli o'zgaruvchiga yuklang.
- (2) o'zgaruvchini `x` ni konsolga chiqaring.

Muammoni quyidagicha hal qilamiz,

- (1) `Salom, Python` qiymatini `x` nomli o'zgaruvchiga yuklang

  ```
  x = 'Salom, Python!'
  ```

- (2) O'zgaruvchi `x` ni konsolga chiqaring

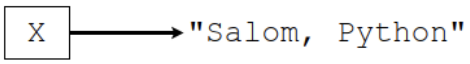
  ```
  print(x)
  ```

1 va 2-ko'rsatmalarni birlashtirsak,

```
x = 'Salom, Python!'
print(x)

--------------
Salom, Python!
```

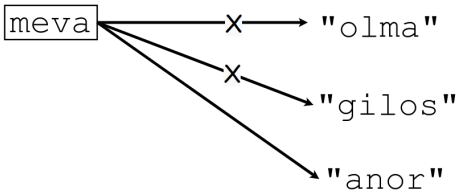
Xullas, `Salom, Python!` qiymatini `x` o'zgaruvchiga yukladik. Keyin `print(x)` yordamizda konsolga chiqardik. Umuman, grafik ravishda quyidagicha tasvirlashimiz mumkin:



Nimaga o'zgaruvchi deb atalishiga misol ko'rsak,

```
meva = "olma"
print(meva)
meva = "gilos"
print(meva)
meva = "anor"
print(meva)

-----
olma
gilos
anor
```

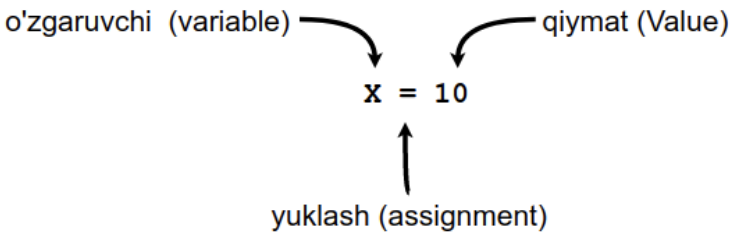


Sonlar bilan ishlasak,

```
x = 10
```

O'zbekchada: `10` qiymatini `x` nomli o'zgaruvchiga yuklang.

Binobarin,



Nomni chiroyli tanlang

Programmistlar o'zgaruvchilarga nom berayotganda ma'noga ega bo'lgan nom berishi kerak. O'sha nomning o'zidan o'zgaruvchi nima uchun foydalanyotganligini bilish oson bo'lsin. Masalan, quyidagi misolda o'zgaruvchi nomi `balandlgi` deb nomlangan va tushunarli.

```
balandligi = 100
```

Agar biz quyidagicha nom bersak, hech kim tushunmaydi,

```
trewqo345235 = 100
```

O'zgaruvchilar nomi uzunligi qanchalik bo'lishligi cheklanmagan. Va ular harf va sonlardan iborat bo'lishligi shart.

```
x = 100
x1 = 200
```

Agar o'zgaruvchi bir necha so'zdan tashkil topsa, u holda `_` belgisini bilan ajratib yozilishligi tavsiya qilinadi -- o'qishga oson (qayiqningbalandligi, qayiq_balanligi),

```
doira_yuzi = 100
uy_balandligi = 80
```

Mumkin bo'lmagan holatlari:

- O'zgaruvchi nomi son bilan boshlanmasligi darkor.
- Ushbu belgilarsiz `@, !, ., ^, *. (, ), $, £, '`

```
1x = 100
@y = 25
```

Agar programmist yuqoridagi qoidalarni bilmasdan o'zgaruvchilarga nom bersa,
quyidagi xatoga duch keladi:

```
SyntaxError: invalid syntax
```

Yana, pythonda **kalit so'zlar** mavjud. Bu so'zlar o'zgaruvchiga nom sifatida foydalanmasligi shart, ya'ni

```
class = "Salom, do'stlar"
```

Aks holda, yana o'sha xato!

```
SyntaxError: invalid syntax
```

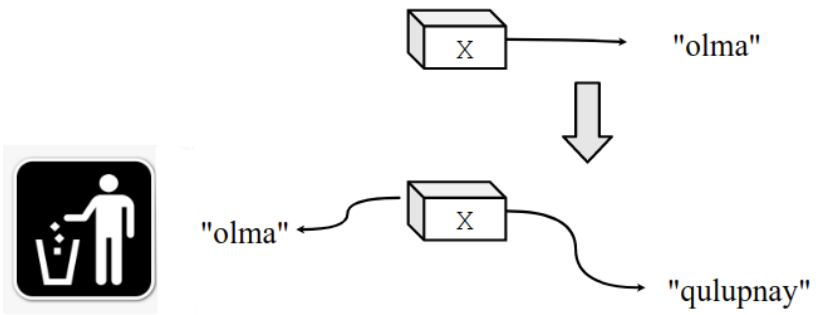
PyCharmda kalit so'zlar alohida rangda ko'rinadi, o'shaning uchun ularni yodlash kerakmas.

Olmami yoki qulupnaymi?

Quyidagi programmani olaylik. Ishlatsak konsolga nima chiqadi deb o'ylaysiz?

```
x = "olma"
x = "qulupnay"
print(x)
```

ekranga `qulupnay` chiqadi. Sababi, yana quti misolida ko'rsak,
ya'ni grafik ravishda quyidagi jarayon bo'layapti,



Avval, qutiga olma yuklandi `x = "olma"`. Keyin qulupnay yuklanmoqchi bo'layapti. Birinchi, olmani olamiz va uni 'musur' ga otamiz.
Keyin, qulupnayni yuklaymiz. Bu qutiga faqat bitta
narsa ketadi. Boshqa narsa yuklansa, oldingi narsa yo'q bo'ladi. Bye-bye olma.

Ifodalar

Shu paytgacha o'zgaruvchilar bilan ishladik. Bu ifodaga eng oddiy misol bo'ladi: bitta o'zgaruvchi va uning qiymati. Biz asta sekin murakkabroq
amallar/ko'rsatmalar bilan ish ko'ramiz. Bunda ifoda nafaqat o'zgaruvchilar va qiymatlar balki operatorlardan ham toshkil topadi. Ifodaga misollar:

```
x = 1
n = 77
y = x + 1
```

- 1-qatorda `1` qiymati `x` ga yuklandi,
- 2-qatorda `77` qiymati `n` ga yuklandi.
- 3-qatorda `x` ga allaqachon 1 yuklangan edi, bunda `1 + 1` bo'ladi. Qo'shishdan keyin `2` bo'ladi. Keyin `2` `y` ga yuklanadi.
  Programma balanddan pastga qatorma-qator ishlaydi. Har bir qatorda esa ifodalar o'ngdan chapga qarab ishlaydi.

Yana bir misol,

```
x = 1
print(x)
```

1-qatorda `x` nomli o'zgaruvchiga `1` yuklandi. 2-qatorda print orqali konsolga chiqarildi.

Operator ketma-ketligi

Ko'pincha ifodada birdan ortiq operatorlar ishtirok etishi mumkin. Bu holat asosan matematik amallarni bajarganda bo'ladi. Masalan,

```
y = 2 + 3**2 * (15 + 3)
```

Operator (amal) lar bajarilish tartibi quyidagicha bo'ladi:

1. qavs eng birinchi bajariladi, yani `(15 + 3) = 18`
2. keyin daraja bajariladi, yani `3**2 = 9`
3. keyin ko'paytirish amali bajariladi, yani `9 * 18 = 162`
4. keyin qo'shish amali bajariladi, yani `2 + 162 = 164`

Xullas, pycharmda hisoblasak

```
y = 2 + 3**2 * (15 + 3)
print(y)

-----
164
```

"String" + "bilan" + "ishlash"

Umuman olganda, stringlar bilan arifmetik amallarni bajarib bo'lmaydi. Masalan, quyidagilar mumkin emas:

```
ayirish: 'kitob' - 'olma'
bo'lish': 'til' / 'oson'
ko`paytirish: 'chaqmoq' * 'alanga'
```

Lekin, quyidagi amallarni bajarish mumkin `+` va `*`.

1. `+` stringlarni bir-biriga birlashtiradi (ketmat ket ulaydi)

   ```
   birinchi = 'oltin'
   ikkinchi = 'baliq'
   natija = birinchi + ikkinchi
   print(natija)

   ----------
   oltinbaliq
   ```

2. `*` stringlarni takrorlab ketma-ket ulaydi

   ```
   salomlar = 'salom' * 3
   print(salomlar)

   ---------------
   salomsalomsalom
   ```

   Yodda tutingki, bunda qiymatlardan biri butun son bo'lish kerak: birinchisi string, keyingisi son yoki o'rni almashsa ham bo'laveradi, ya'ni

   ```
   salomlar = 3 * 'salom'
   print(salomlar)

   -----------------------------------------------
   salomsalomsalom # 3 marta `salom` ni takrorlaymiz
   ```

Izoh qoldirish

Tabiiyki, agar programma murakkabroq bo'lsa ko'p ifodalarni yozishga to'gri keladi. Bu o'z navbatida kodni murakkabligini oshiradi. Mana shu maqsadda kodning malum joyi nimaga unday yozilganligini bildirish uchun **izoh** qoldirishimiz mumkin. Foydaliligi,

- O'zimiz uchun. O'zimizning esimizdan chiqmasligi uchun, 1 haftadan keyin kelib kodga qarasak kod nima uchun yozilganligi esimizdan chiqishi mumkin.
- Boshqa programmistlar uchun. Biz yozgan kodni boshqa odamlar ham foydalanishi mumkin. Ularga kodni nima maqsadda yozilganligini bildirish uchun.

Izoh inglizchada **comment** deb ataladi. Bundan buyon komment deb ketamiz.
Kommentlar programmada `#` belgisi bilan yoziladi. Quyida, kommentga misollar kettirilgan.

```
# string: ko'paytirishga misol
salomlar = 'salom' * 3
print(salomlar)

salomsalomsalom # 3 marta `salom` ni takrorlaymiz, chunki yuqorida 3 ga
ko'paytirdik
```

Komment qo'yayotganda uning kerak yoki kerakmasligiga e'tibor berish kerak.
Masalan, quyidagi komment foydasiz, chunki kommentsiz ham biz nima bo'layotganini fahmlab olishimiz juda oson.

```
x = 11 # 11 soni x ga yuklanayapti
```

Foydali kommentga misol sifatida, quyidagini keltirishimiz mumkin. Yani, biz `x` ning qanday o'zgaruvchi
ekanligini aytib o'tayapmiz. Keyinchalik, bu o'zgaruvchi nima uchun kerak edi deyishdan saqlanish uchun!?

```
x = 11 # quti bo'yi (m)
```

Xo'sh, o'zgaruvchi yana nima uchun kerak?

Aytaylik quyidagi ifoda berilgan,

```
2 + 3 * (10**2 + 2 / 0.6)
```

Bir narsa bo'ldiki biz ifodadagi `2` ni `5` soniga o'zgartirishimiz kerak. Unda nima qilamiz? Bizda hozircha ikki yo'l mavjud, jumladan,

- To'g'ridan to'g'ri: har bir `2` ning o'rniga `5` ni yozamiz

  ```
  5 + 3 * (10**5 + 5 / 0.6)
  ```

- O'zgaruvchi e'lon qilamiz va `5` ni unga yuklaymiz. Keyin ifodaga o'zgaruvchini qo'yamiz.

  ```
  x = 5
  natija = x + 3 * (10**x + x / 0.6)
  ```

  Chinakkam programmist bo'laman degan inson 2-chi yo'ldan ketish kerak. Chunki, u yana va yana
  qayta va qayta yozaverishni xoxlamaydi. Muhimi, kod tushunarli va chiroyli va tartibli bo'ladi. Qolaversa, endi xoxlagan songa bir zumda o'zgartirish mumkin: faqat birinchi qatorni o'zgartiramiz bo'ldi `x = 10`, `x = 25` va hokazo.

3 xil xato qilishingiz mumkin

Programma tuzish mobaynida biz 3 xil xato qilishimiz mumkin.

1. Sintaks (imploviy) xato. Bunda biz python qoidasidan chiqib yozgan bo'lishimiz mumkin. Masalan, o'zgaruvchiga nom
   xarflardan tashkil topish kerak, lekin biz `@x = 1` deb yozgan bo'lishimiz mumkin. Bu huddi, ona tilimizdagi

kabi xato. Masalan, `kitob` so'zini `kitib` deb yozib bo'lmaydi - xato hisoblanadi.

2. Ishlash mobaynidagi (runtime) xato. Programmada sintaks xato bo'lmaydi. Faqat programma to'liq ishlamaydi --
   qisman ishlab xato berib yuboradi. Bunday xatolar programmada noodatiy hodisa ro'y berganini
   bildiradi. Masalan, quyidagi programmada hech qanday sintaks xato yo'q, lekin sonni nolga bo'lish mumkin
   emas. Bu qoidani esdan chiqarishimiz mumkin

   ```
   x = 1/0
   ```

3. Semantik xato. Bunda programma ishlaydi. Na runtime xato na sintaks xato bor. Lekin, programma natijasi siz
   xoxlagan natijani bermaydi. Xatolar ichida ushbu tur eng murakkabi. Xatoni topish uchun programmani yana
   qaytadan ko'rib chiqish kerak -- haqiqatdan ham algoritm to'g'riligini tekshirish kerak. Bunda debugging mahorati qo'l keladi.

Xullas, 1- va 2- xatolarni topishda pycharm yoki python interpretatori yo'l yo'riq ko'rsatadi. 3-xatoda
esa programmistning o'zi algortmni (ko'rsatmalarni) xato tuzgan bo'ladi. Kompyuterda ayb yo'q, hamma ayb sizda.

Foydali terminlar

- **o'zgaruvchi** (variable): biron qiymatni yuklash uchun kerak bo'ladigan joy.
- **yuklash** (assignment): `=` yuklashni amalga oshirish belgi.
- **kalit so'z** (keyword): faqat python tili uchun programmistlar tomonidan qo'yilgan nomlar.
- **ifoda** (expression): Biron natija olish uchun birlashtirilgan o'zgaruvchilar, operatorlar va qiymatlar.
- **ishlatish** (exucute): kodni ishlatish.
- **birlashtirish** (concatenate): stringlarni birlashtirish.
- **kommment** (comment): programmadagi foydali izoh. Bu programmist uchun, programma uchun emas. Agar boshqa programmistlar kodni ko'rsa tez tushunishliligi uchun. Programmada kommentlar ishlamaydi.

Problem solving

1. O'rganganimizdek, o'zgaruvchilar quyidagicha elan qilinadi, `x = 12`. Agar. `12=x` qilinsa nima bo'ladi?
2. `x = 1/0` ishlating va natijani tushuntiring?
3. `x = y = z = 111` ishlating va `x, y, z` larni ekranga chiqaring.
4. `x = 5;` ishlating va natijani tushuntiring?
5. `x = 1` va `y = 2` ikkita o'zgaruvchni ko'paytiring, bo'ling, qo'shing, ayiring, darajaga oshiring Natijalarni ekranga (konsolga) chiqaring.
6. `a = 2, b = 1, c = 5` quyidagini hisoblang va konsolga chiqaring

$$X = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

7. 27 ning 15% ni toping, natijani 'miqdor' ga yuklang.
8. 30 ni .25 ko'paytiring va natiajni `x` ga yuklang.
9. Konsolga `Assalomu Alaykum` deb 100 marta chiqaring.
10. `s1="1"` va `s2="2"`. `z = s1 + s2`? `z = s2 + s1` ? `z = s2 * s1`? `z = s2 - s1`?
11. Quyigani ishlatish va tushuntiring?

```
s = "salom"
s = s + s
```

12. Konsolga `oltin baliq` deb deb chiqaring.
13. Konsolga 'salom salom salom' deb chiqaring.
14. Operator bilan qiymatning nima farqi bor?
15. Quyidagilardan qaysi biri o'zgaruvchi va qaysi biri string?

```
yashik
'yashik'
```

16. o'zgaruvchi turlarini sanang?
17. Man bu programmani ishlating. Xato bo'lsa to'g'irlang.

```
print(2 + '2')
```